
I2C

SPI

3-Wire

comunicación serie **sincrónica**

I2C / IIC

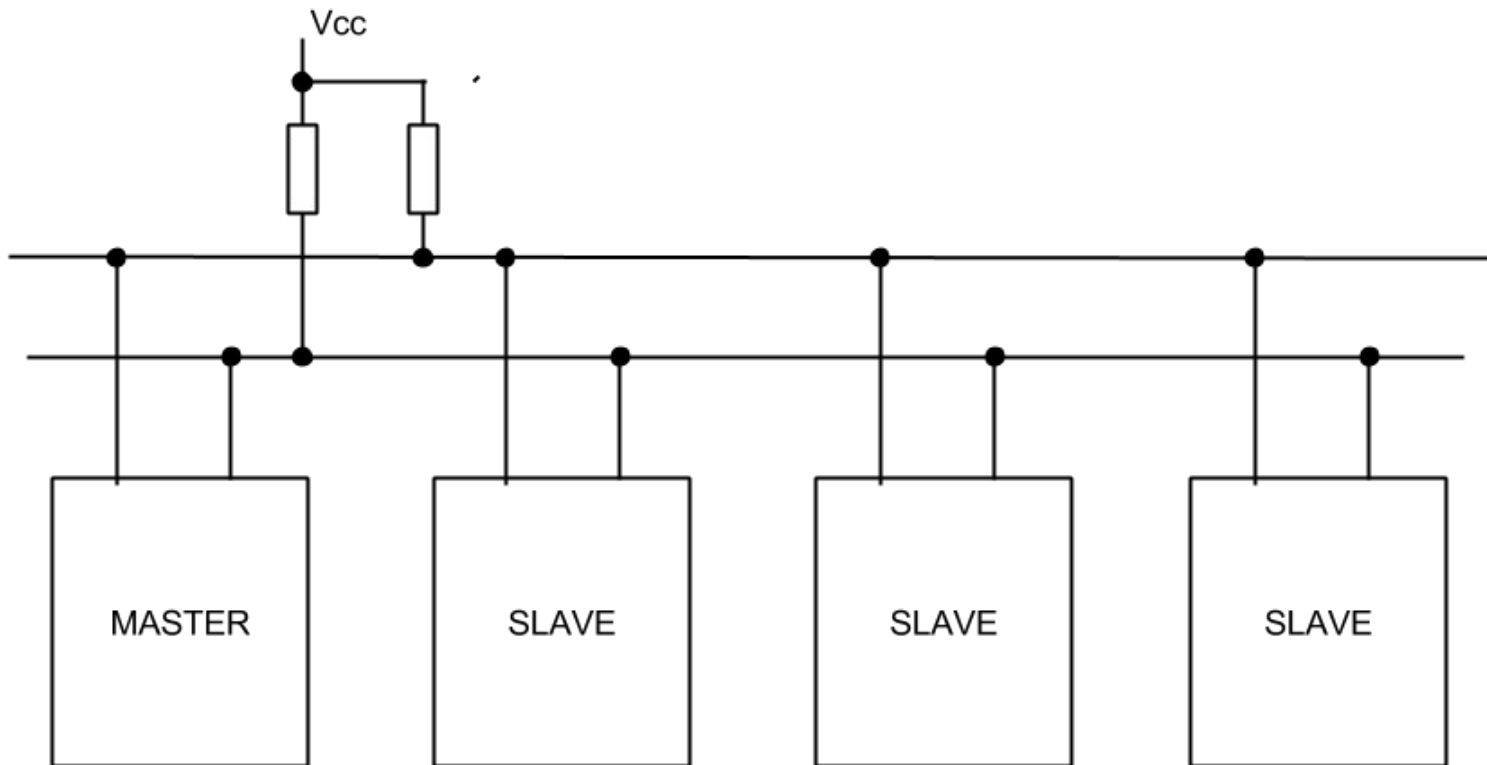
I2C - conexión

- La interconexión se realiza usando dos cables:
 - SDA - datos
 - SCL - clock
 - Los dispositivos I2C se conectan a estos cables (al bus I2C) con open-collector / open-drain.
 - Clock: standard hasta 100 KHz - Fast: 400 KHz - High speed 3.4 MHz
-

I2C - conexión

- Hay que colocar **dos** resistores de pull-up (Entre 1K y 10K, uno en SCA y otro en SCL)
 - El “cero” en el bus se obtiene si al menos un dispositivo se coloca en estado bajo.
 - Y el “uno” cuando todos los transistores de salida están al corte (por eso, **sin Rpu no funciona**)
-

I2C - conexión



I²C

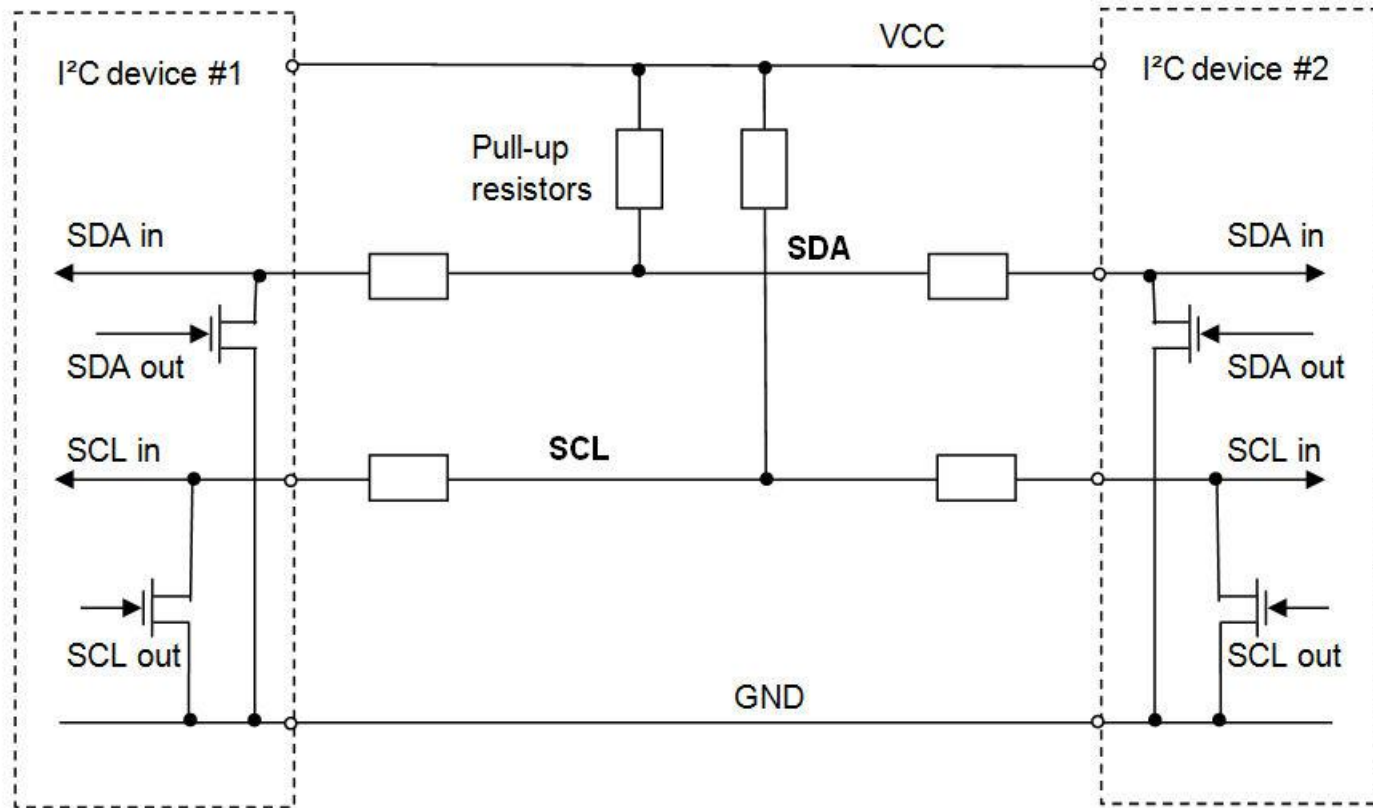
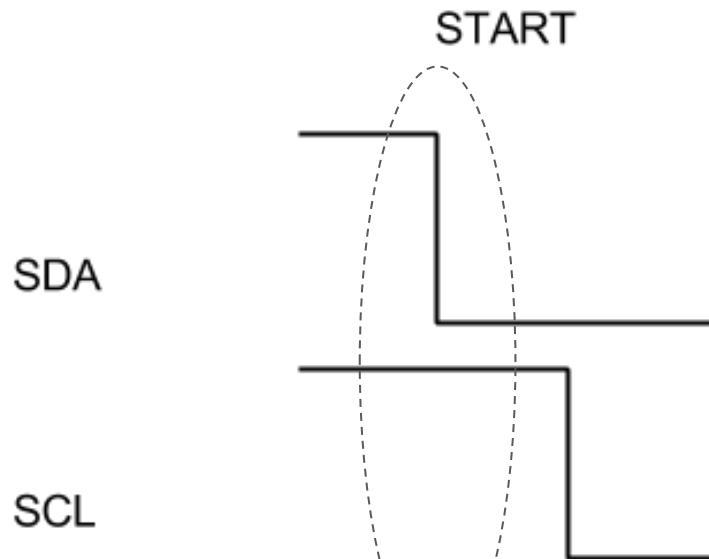


Figure 4: I²C bus with 2 devices connected. SDA and SCL are connected to VCC through pull-up resistors. Each device controls the bus lines outputs with open drain buffers.

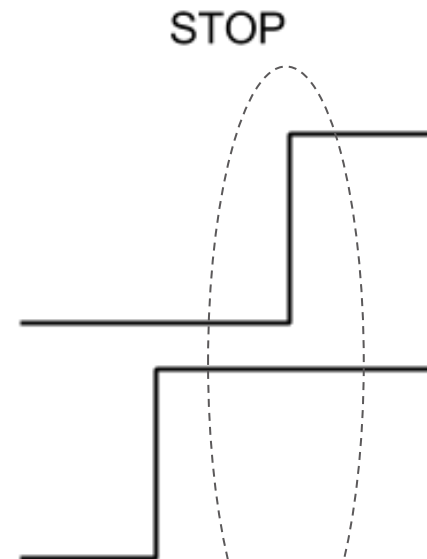
I2C - protocolo físico

- Cuando se transmiten datos / comandos SDA tiene que permanecer estable (0 o 1)
 - Si SDA cambia y SCL no, es una de dos condiciones especiales:
 - start: el master inicia comunicación.
 - stop: el master da por terminada la transacción.
-

I2C - protocolo físico



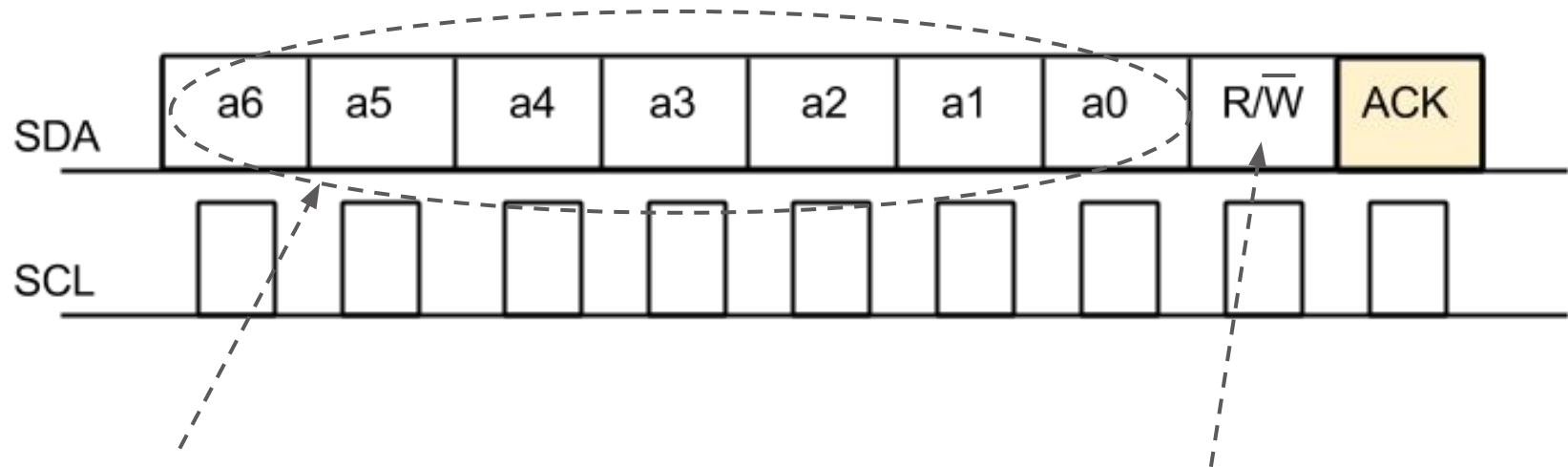
SCL -> sin cambios
SDA -> flanco descendente



SCL -> sin cambios
SDA -> flanco ascendente

I2C - direccionamiento

- Para identificar a qué dispositivo del bus se le habla y si se quiere leer o escribir:

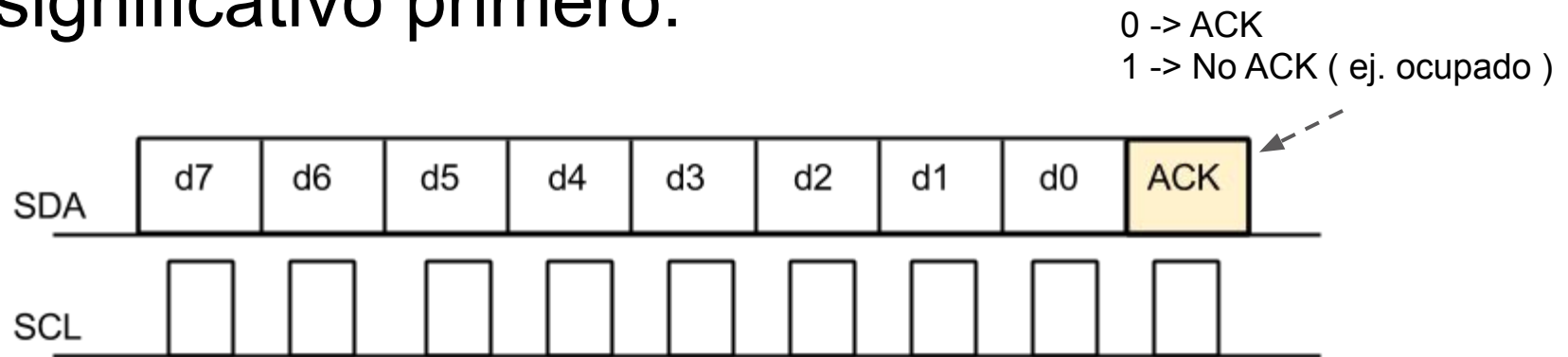


a6..a0 -> hasta 128 direcciones diferentes.

0: se va a **escribir**
1: se va a **leer**

I2C - protocolo físico

- Los datos se envían con el bit más significativo primero:



- El dispositivo que recibe, "responde" (mantiene SDA en uno o cero) durante el noveno ciclo de clock: bit de Acknowledge.

I2C - protocolo lógico

Para escribir:

1. El master envía **start**
2. El master envía ID de dispositivo + **w**
3. El master envía registro a escribir
4. El master envía datos (8 bits)

El master envía más datos...

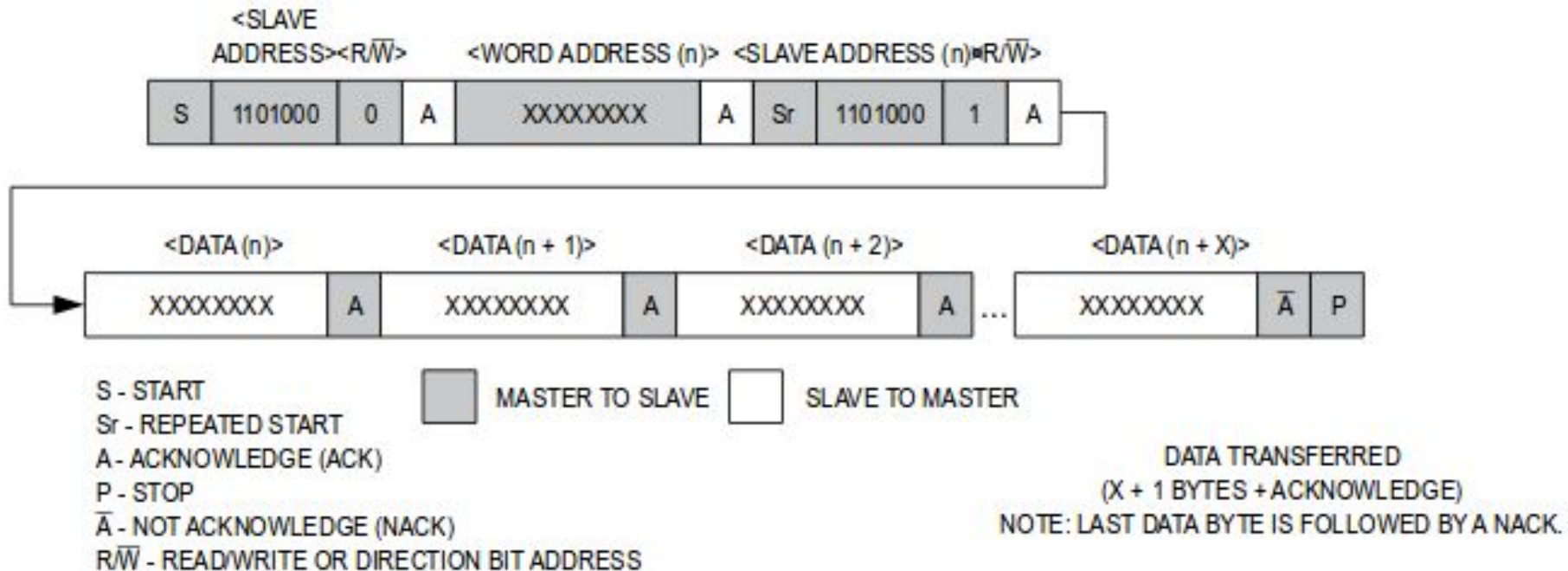
5. El master envía **stop**
-

I2C - protocolo lógico

Para leer:

1. El master envía **start**
 2. El master envía ID de dispositivo + **w**
 3. El master envía registro a leer
 4. El master envía **start** (corta operación de escritura)
 5. El master envía ID de dispositivo + **r**
 6. El slave envía **datos** (8 bits)
 7. El master envía **stop** para finalizar operación o ACK para que el slave envíe otro byte de datos.
-

I2C - protocollo



Data Write/Read (Write Pointer, Then Read) — Slave Receive and Transmit

(En un RTC DS3231)

I2C - protocollo

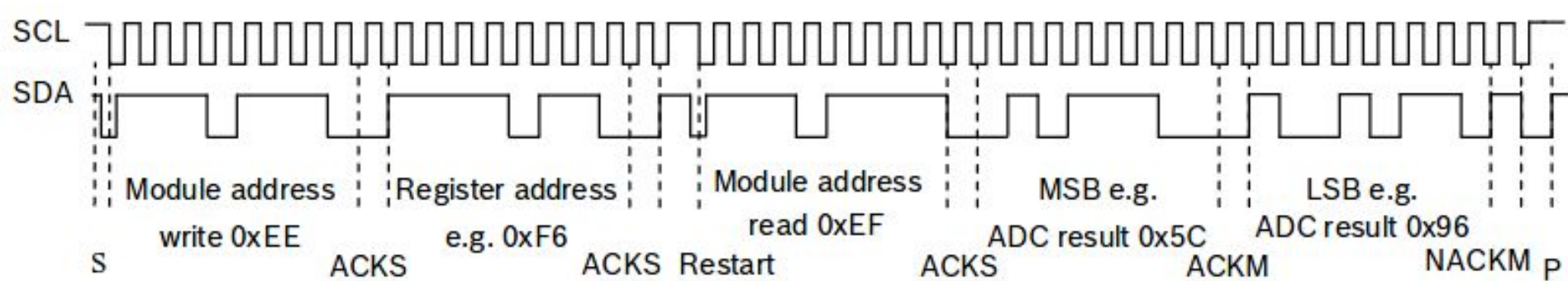
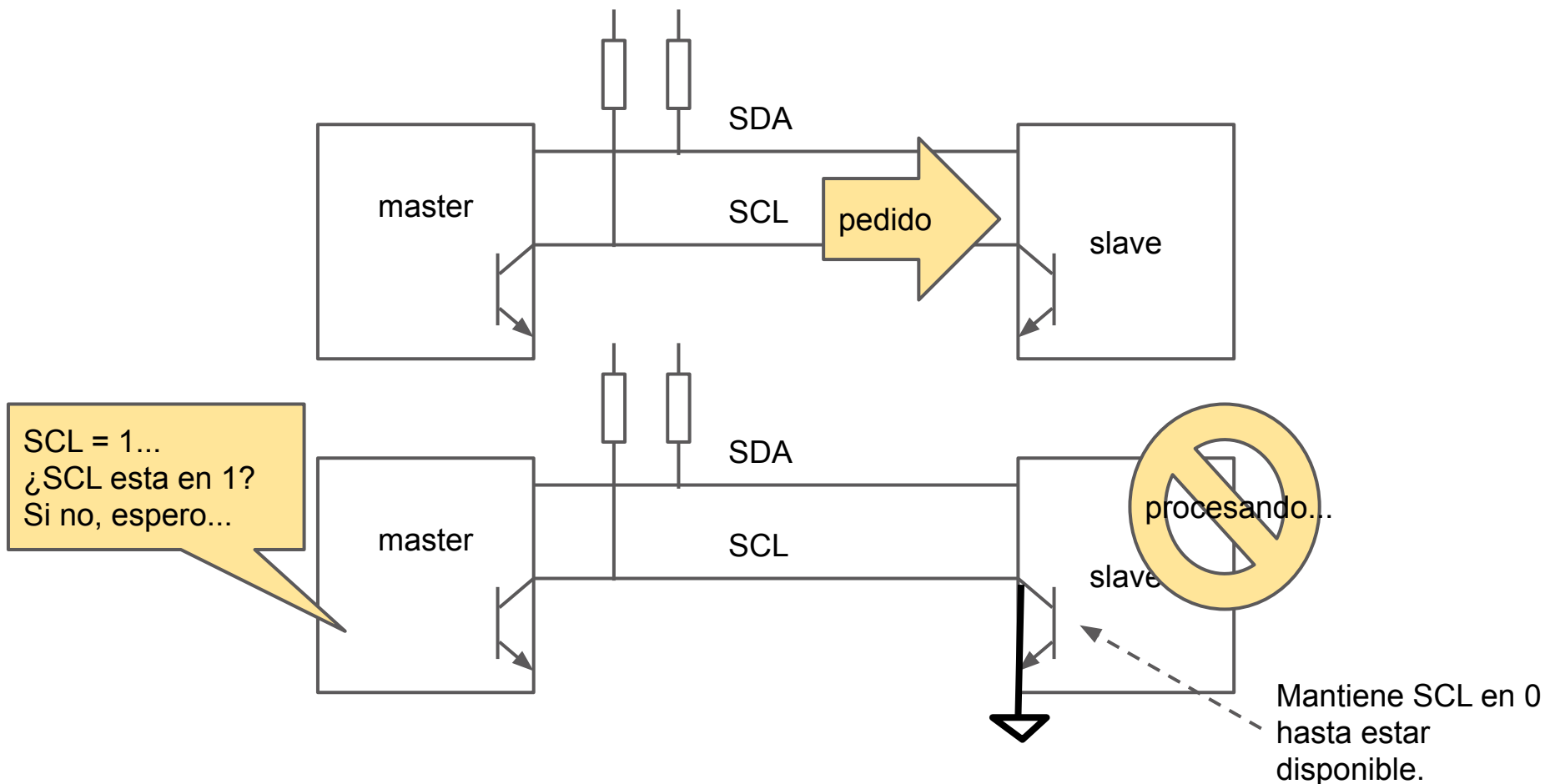


Figure 9: Timing diagram read 16 bit A/D conversion result

(En un BMP180)

I2C - clock stretching

El master tiene que esperar para generar clocks a que el slave termine de procesar...



I2C - protocolo - primitivas ejemplo

```
void i2c_start(void) {  
    SDA = 1;  
    i2c_dly();  
    SCL = 1;  
    i2c_dly();  
    SDA = 0;  
    i2c_dly();  
    SCL = 0;  
    i2c_dly();  
}
```

```
void i2c_stop(void) {  
    SDA = 0;  
    i2c_dly();  
    SCL = 1;  
    i2c_dly();  
    SDA = 1;  
    i2c_dly();  
}
```

I2C - protocollo

```
unsigned char i2c_rx(char ack) {
    char x, d = 0;
    SDA = 1;
    for(x = 0; x < 8; x++) {
        d <<= 1;
        do {
            SCL = 1;
        } while(SCL_IN==0); // SCL ck stretching?
        i2c_dly();
        if(SDA_IN) d |= 1;
        SCL = 0;
    }
    if(ack)
        SDA = 0;
    else
        SDA = 1;
    SCL = 1;
    i2c_dly();
    SCL = 0;
    SDA = 1;
    return d;
}
```

I2C - protocollo

```
unsigned char i2c_tx(unsigned char d) {
    char x;
    unsigned char b;

    for(x=8; x; x--) {
        if(d&0x80)
            SDA = 1;
        else
            SDA = 0;
        SCL = 1;
        d <<= 1;
        SCL = 0;
    }
    SDA = 1;
    SCL = 1;
    i2c_dly();
    b = SDA_IN; // ACK de slave?
    SCL = 0;
    return b;
}
```

SPI

SPI

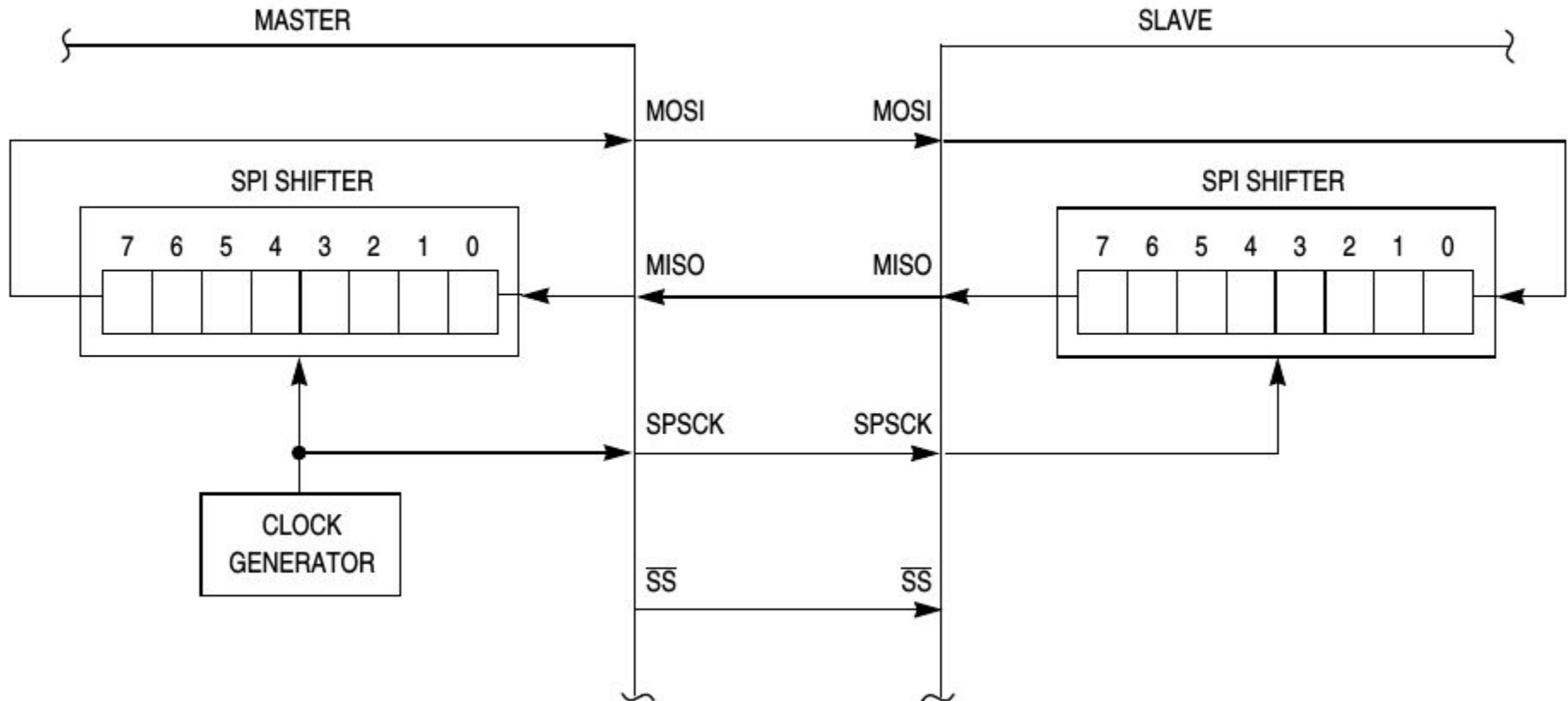
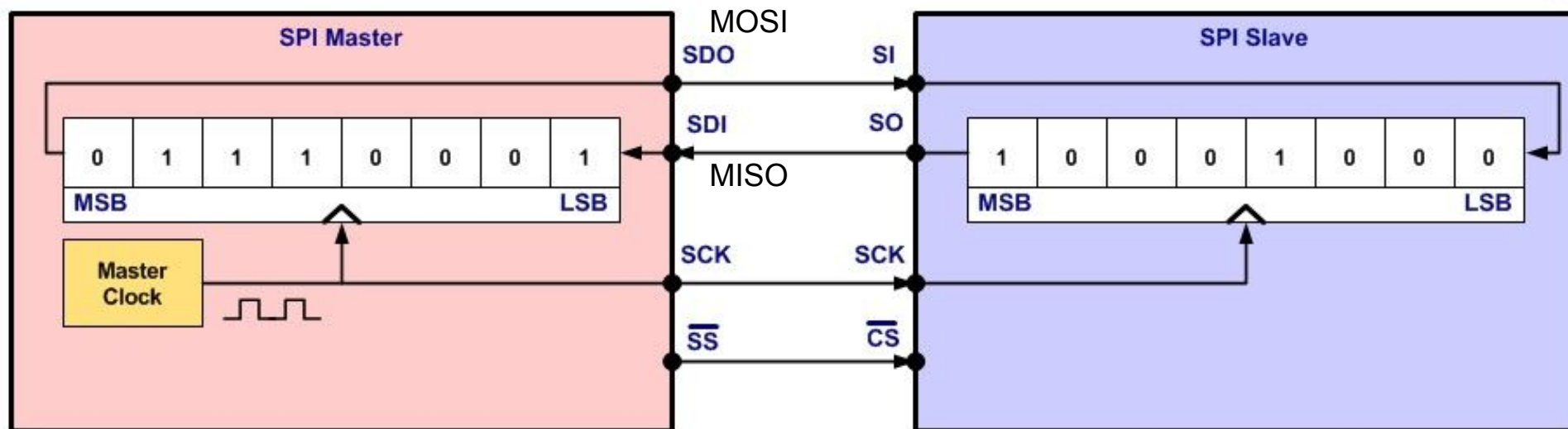


Figure 15-2. SPI System Connections

SPI

<http://www.ermicro.com/blog>

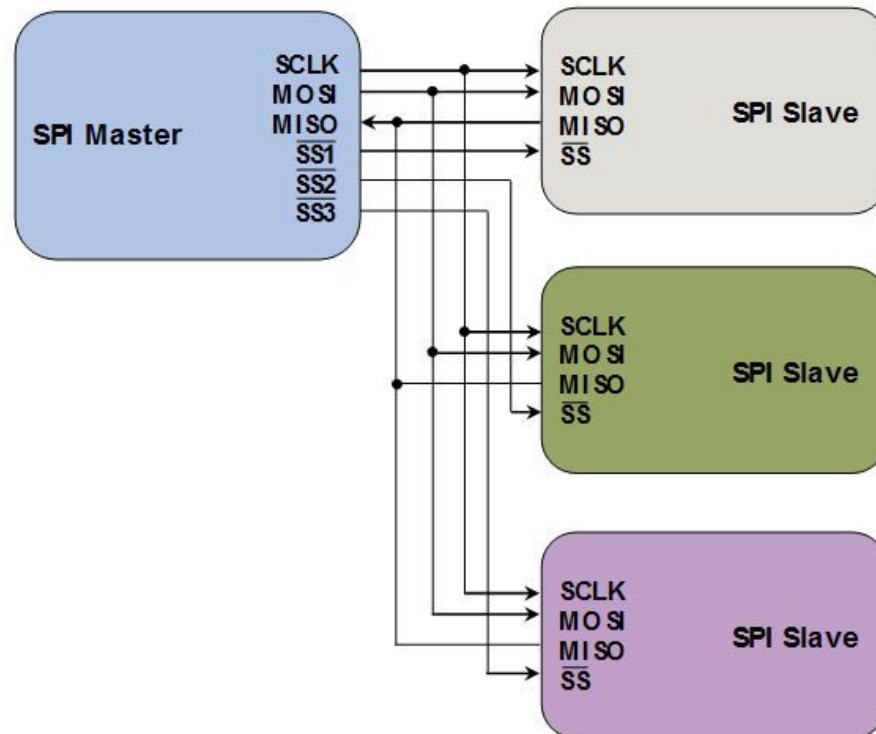
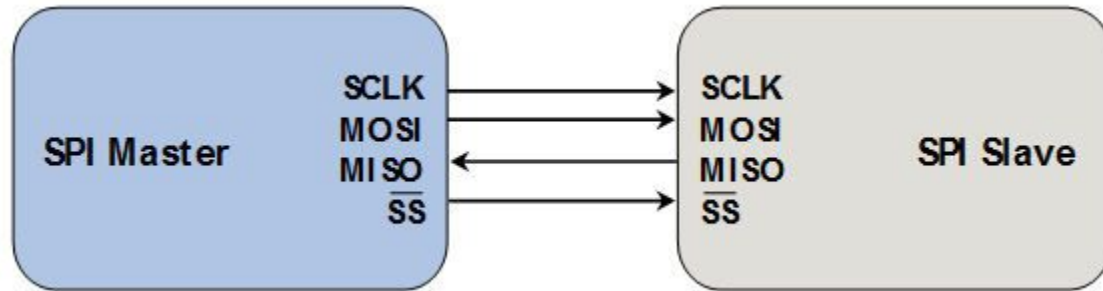


PIN	Description	Direction
SDO	Serial Data Out, This pin also called as MOSI (Master Out Serial In) in AVR Type Microcontroller	Out
SDI	Serial Data In, This pin also called as MISO (Master In Serial Out) in AVR Type Microcontroller	In
SCK	Serial Clock (SPI Master)	Out
SS	Serial Select (could be any I/O Port)	Out

PIN	Description	Direction
SI	Serial In	In
SO	Serial Out	Out
SCK	Serial Clock (SPI Slave)	In
CS	Chip Select (usually active low)	In

The SPI Master and SPI Slave Basic Connection

SPI



SPI

- La interconexión se realiza usando tres cables:
 - MOSI - Master Out - Slave In
 - MISO - Master In - Slave Out
 - SCL - Serial CLock
 - + SS - Slave Select (1 x cada slave)
 - El master genera clock y selecciona (con SS) el slave al que se le habla.
 - El clock no tiene que ser preciso
 - Velocidad Mbits / s
-

SPI

- El protocolo lo define la aplicación
- Configuración:

CPHA (clock phase) - clock vs dato, quién va primero

CPOL (clock polarity) - en qué estado está el clock cuando está inactivo.

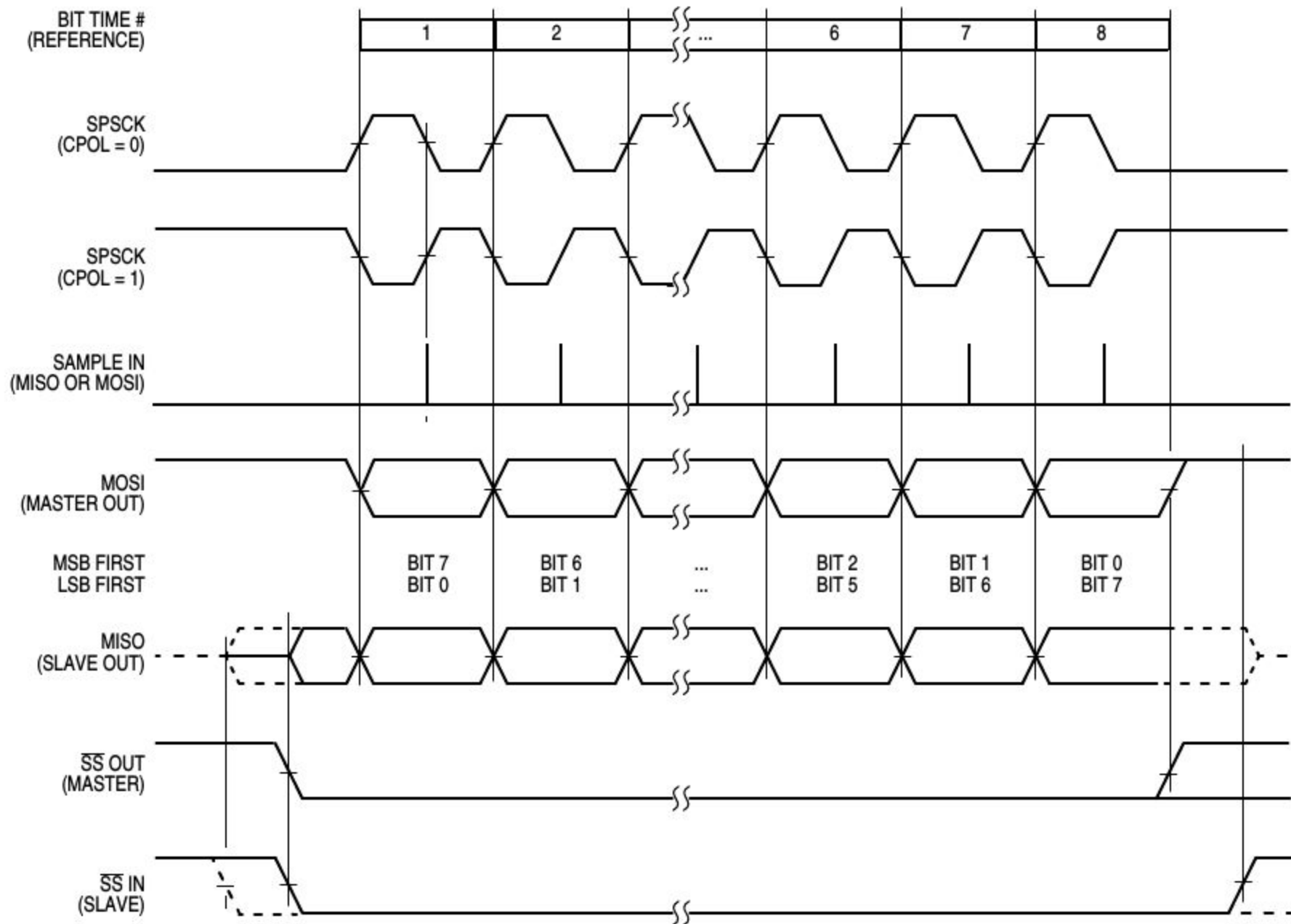


Figure 15-10. SPI Clock Formats (CPHA = 1)

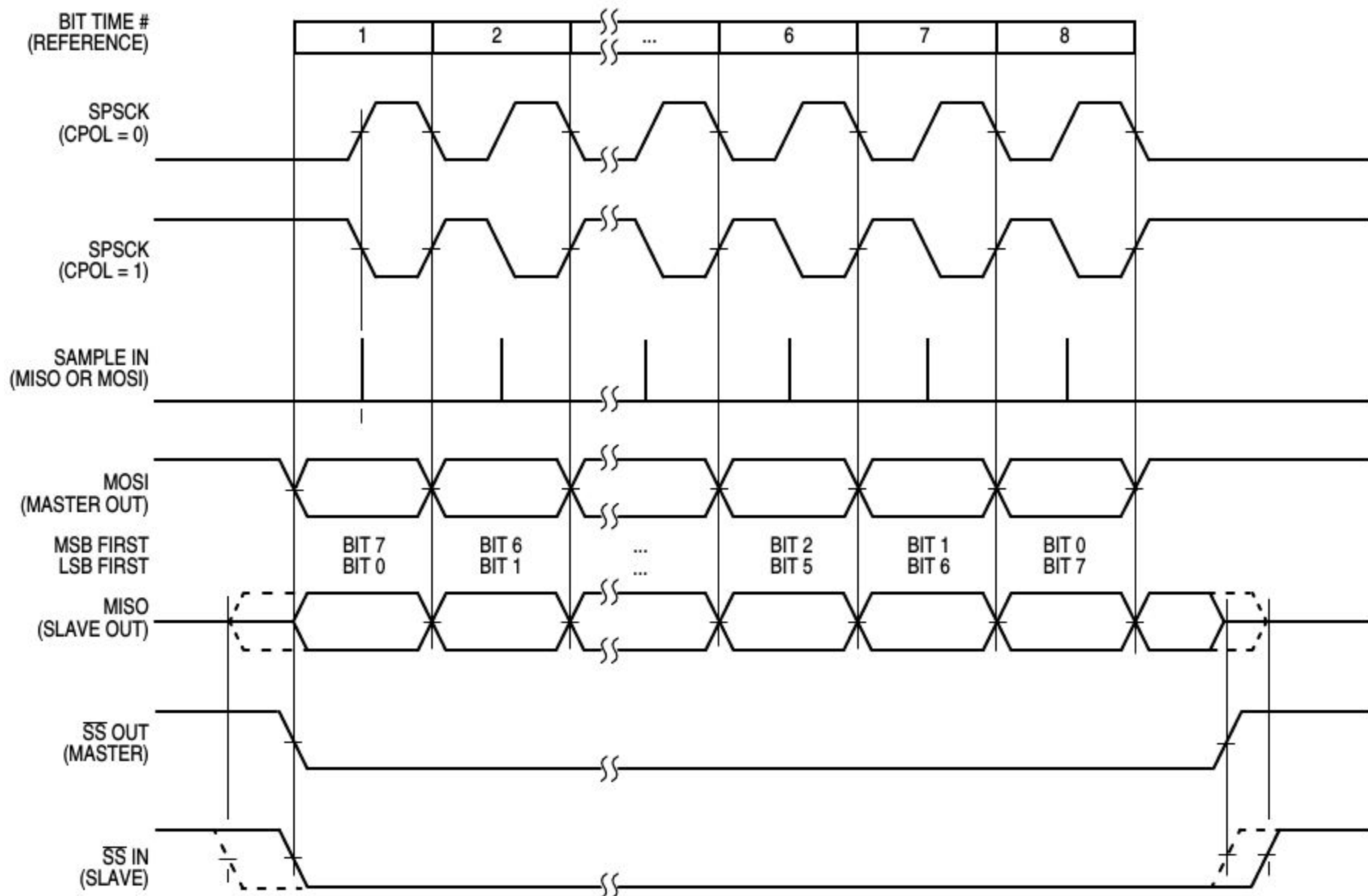
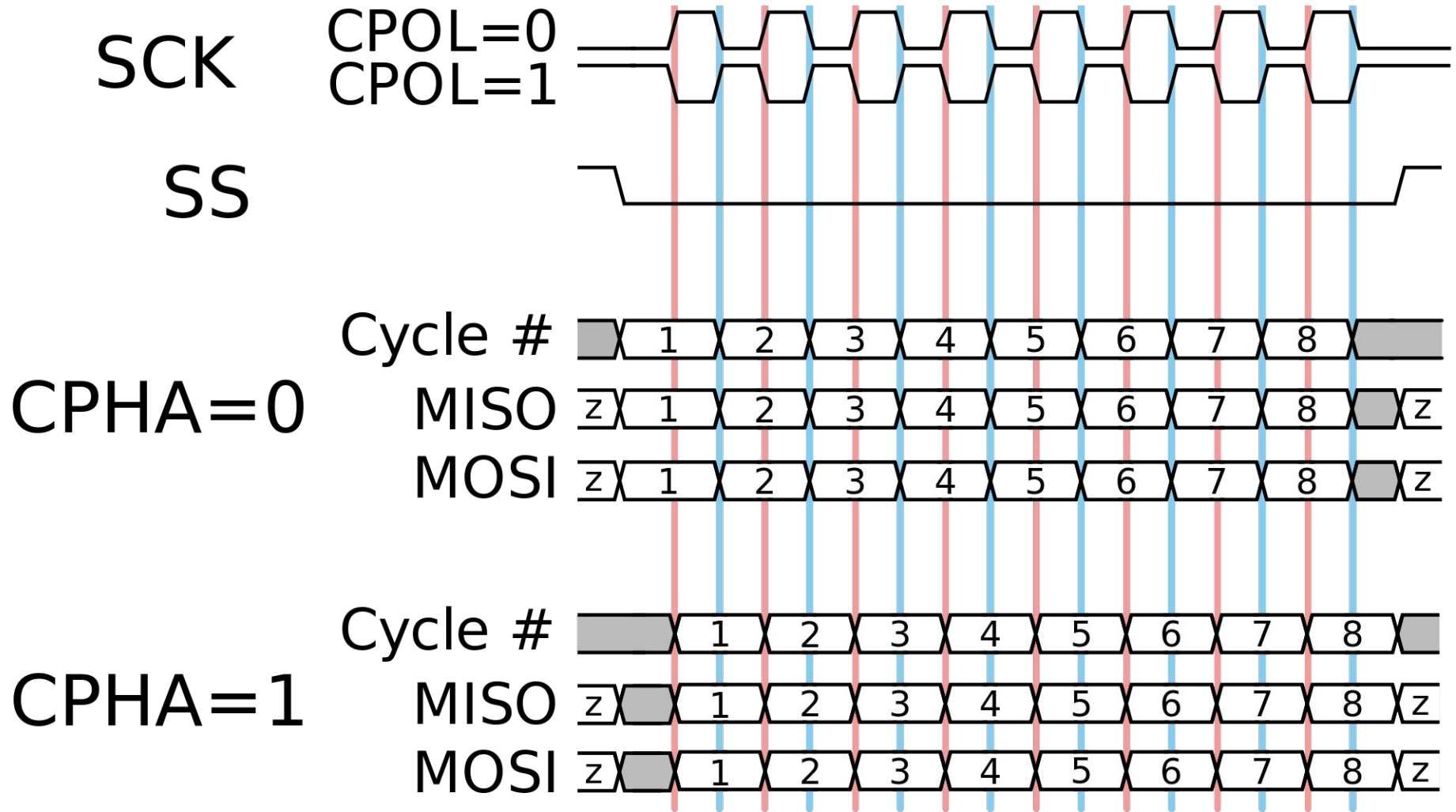


Figure 15-11. SPI Clock Formats (CPHA = 0)

SPI



SPI

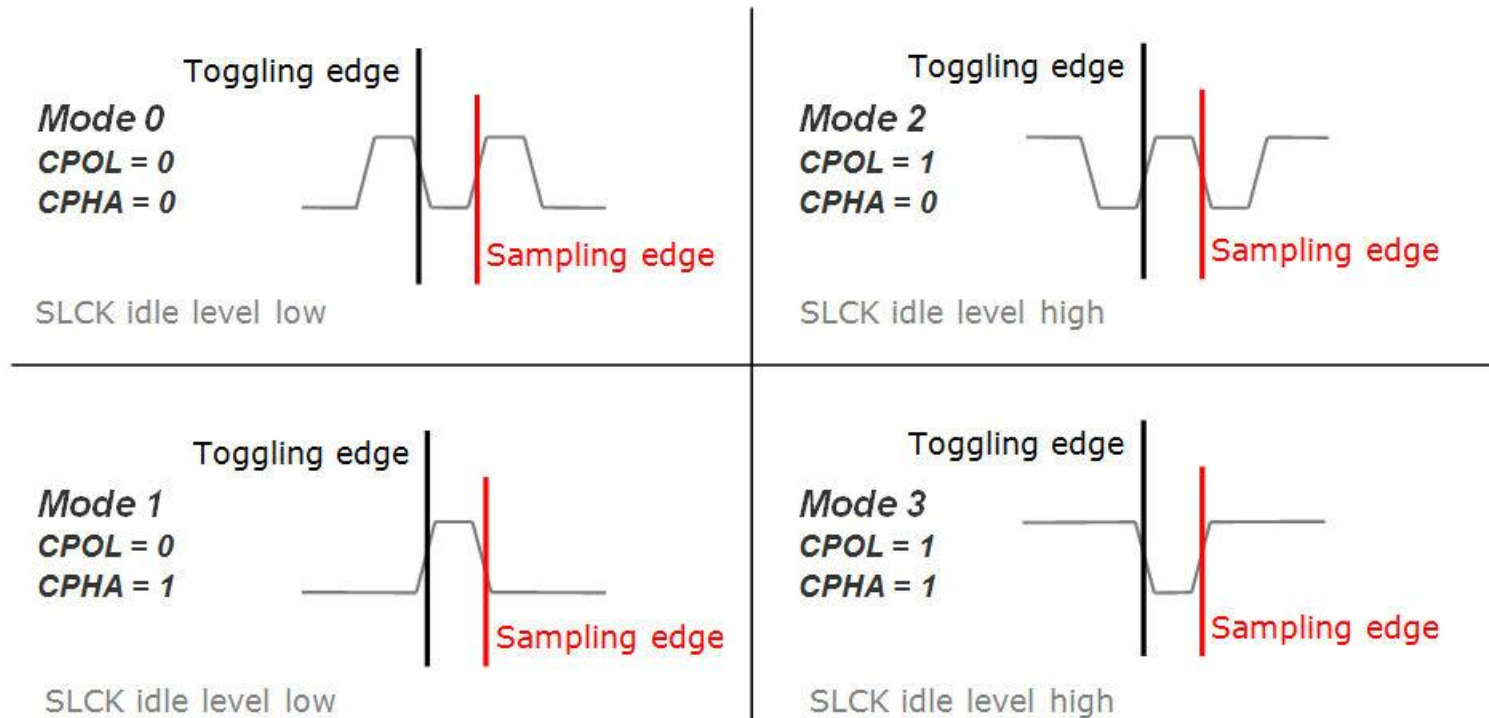


Figure 3 : SPI modes are defined with the parameters 'CPOL' – clock polarity and 'CPHA' – clock phase, which explicitly define 3 parameters: the edges used for data sampling and data toggling and the SCL clock signal idle level – that is the conventional level SCLK is set at when the bus is not in communication.

3-Wire

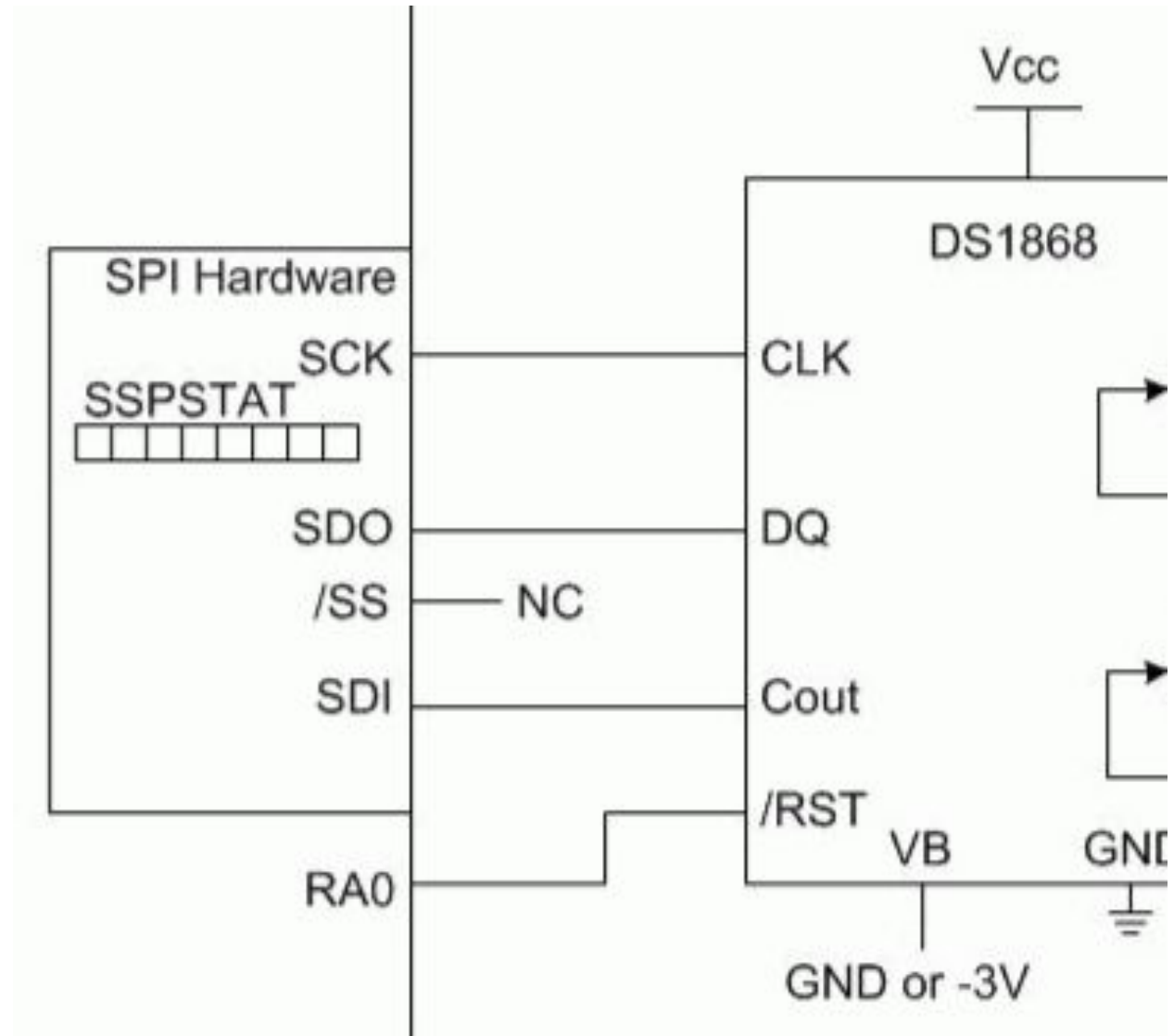
3-Wire

- No es SPI
- Pero es Compatible con SPI

Table 1. 3-Wire vs. SPI Interface.

Serial Interface	Signal names	Signal Descriptions	BUS Speed (Typical)	Data Format
3-WIRE	DQ	Data In	500kHz to 5 MHz	LSB first, MSB last
	Cout	Data Out		
	active-low RST	active-low Reset		
	CLK	Clock		
SPI	SDI	Data In	10MHz	MSB first, LSB last
	SDO	Data Out		
	active-low SS	active-low Slave Select		
	SCK	Clock		

3-Wire



3-Wire
